

Finitary semantics of linear logic and higher-order model-checking

Charles Grellois and Paul-André Melliès

Laboratoire PPS, Université Paris Diderot, Sorbonne Paris Cité
 {grellois,mellies}@pps.univ-paris-diderot.fr

Abstract. In this paper, we explain how the connection between higher-order model-checking and linear logic recently exhibited by the authors leads to a new and conceptually enlightening proof of the selection problem originally established by Carayol and Serre using collapsible pushdown automata. The main idea is to start from an infinitary and colored relational semantics of the λY -calculus already formulated, and to replace it by its finitary counterpart based on finite prime-algebraic lattices. Given a higher-order recursion scheme \mathcal{G} , the finiteness of its interpretation in the model enables us to associate to any MSO formula φ a new higher-order recursion scheme \mathcal{G}_φ resolving the selection problem.

Keywords: Higher-order model-checking, linear logic, selection problem, finitary semantics, parity games.

1 Introduction

Higher-order recursion schemes (HORS) provide an abstract model of computation which appears to be perfectly adapted for the task of model-checking functional programs. Indeed, Knapik, Niwinski and Urzyczyn established in [7] that for $n \geq 1$, the trees generated by order- n safe recursion schemes are exactly those that are generated by order- n pushdown automata, and further, that they have decidable MSO theories. The MSO-decidability result for safe HORS was then extended a few years later to all HORS by Ong [9]. However, the MSO-decidability theorem established by the four authors focuses on the decidability of a “local” model-checking problem:

Suppose given a HORS \mathcal{G} which generates an infinite tree $\langle \mathcal{G} \rangle$. Is it possible to decide for every MSO-formula φ whether the formula is valid at the root of the infinite tree $\langle \mathcal{G} \rangle$.

The MSO-decidability result means that the answer to this question is positive. A more difficult “global” model-checking problem called the *selection problem* in literature is to understand whether:

Given a HORS \mathcal{G} and a MSO-formula $\exists X \varphi[X]$ holding at the root of the infinite tree $\langle \mathcal{G} \rangle$, is it possible to compute a HORS \mathcal{G}_φ generating a marked version $\langle \mathcal{G}_\varphi \rangle$ of the original tree $\langle \mathcal{G} \rangle$, and such that the set of its marked nodes is a witness U satisfying the MSO-formula $\varphi[X]$.

Quite strikingly, Carayol and Serre established in a recent paper [2] that the answer to this question is positive. They also noticed that the selection problem follows from a purely automata-theoretic property of HORS, which was established by Haddad in his PhD thesis [6]:

Given a HORS \mathcal{G} and an alternating parity tree automaton \mathcal{A} with the same ranked alphabet, for every state q of the automaton \mathcal{A} accepted by the tree $\langle \mathcal{G} \rangle$, it is possible to compute a HORS \mathcal{G}_q generating an accepting run-tree $\langle \mathcal{G}_q \rangle$ of the automaton \mathcal{A} on the tree $\langle \mathcal{G} \rangle$ with initial state q .

Of course, the run-tree $\langle \mathcal{G}_q \rangle$ generated by the HORS \mathcal{G}_q provides a witness of the fact that the state q is accepting. But not only that: thanks to the equivalence between MSO-formulas and alternating parity tree automata, the fact that the HORS \mathcal{G}_q selects a *specific* run-tree $\langle \mathcal{G}_q \rangle$ among all the run-trees with initial state q provides a solution to the “selection problem”. The idea is simply to extract from the run-tree $\langle \mathcal{G}_q \rangle$ a specific witness X for the MSO-formula $\exists X \varphi[X]$ satisfied by the tree $\langle \mathcal{G} \rangle$.

In this article, we will show how to establish the existence of such a “higher-order recursive” run-tree $\langle \mathcal{G}_q \rangle$ from purely denotational arguments, based on a new and fundamental connection with linear logic developed by the authors in a series of recent papers [4,5]. In these papers, an infinitary and colored variant of the traditional semantics of linear logic is constructed, see [4] for details, and shown to compute in a compositional way the set of accepting states of an alternating parity tree automaton, see [5] for details. Despite the conceptual clarification this approach provides to higher-order model-checking, this semantic account does not lead to any decidability result. The reason is that the relational semantics of linear logic is a *quantitative* semantics, where finite types are interpreted as infinitary objects. In order to establish decidability results, one thus needs to shift to *qualitative* semantics where the interpretation of finite types remains finite. This is precisely the purpose of the present paper: by shifting from the relational semantics developed in [4,5] to the qualitative semantics of linear logic provided by prime-algebraic lattices, we are able to establish advanced decidability results like the theorem just mentioned by Carayol, Haddad and Serre. This is the first time, to our knowledge, that such a strong and natural connection between model-checking and the most contemporary tools of semantics (linear logic, relational semantics) is exhibited.

Plan of the paper. We start by recalling in §2 the notion of higher-order recursion scheme and its correspondence with the λY -calculus. We then recall in §3 the notion of alternating parity tree automaton. In §4, we introduce a *finitary* colored semantics of the λY -calculus, which we use in §5 to interpret λ -terms. We define a parameterized fixpoint in this model in §6, obtaining colored semantics of the λY -calculus. In §7, we use the finiteness of the model to prove the decidability of the local model-checking and of the selection problem. We finally conclude in §8.

2 Higher-order recursion schemes and the λY -calculus

Higher-order recursion schemes. The set of simple types of the λ -calculus is generated by the grammar $\sigma, \tau ::= o \mid \sigma \rightarrow \tau$. We write $t :: \sigma$ when a (possibly open) λ -term t has simple type σ . Given a ranked alphabet Σ , a finite set of variables \mathcal{V} , a finite set of simply-typed *non-terminals* \mathcal{N} , and a distinguished non-terminal $S \in \mathcal{N}$, a higher-order recursion scheme (HORS) is the data, for every non-terminal $F \in \mathcal{N}$, of a closed simply-typed λ -term

$$\mathcal{R}(F) = \lambda x_1. \dots \lambda x_n. t \quad (1)$$

of same type as the non-terminal $F \in \mathcal{N}$, with constants in Σ , where $x_i \in \mathcal{V}$ and $t :: o$ is a λ -term of ground type without λ -abstractions. Note that an element $a \in \Sigma$ of arity n is represented as a constant of type $o \rightarrow \dots \rightarrow o \rightarrow o$ with same arity n . For each non-terminal $F \in \mathcal{N}$, the data provided by $\mathcal{R}(F)$ is equivalently represented as a rewrite rule

$$F t_1 \dots t_n \rightarrow_{\mathcal{G}} t[x_i \leftarrow t_i].$$

Every higher-order recursion scheme \mathcal{G} generates a potentially infinite Σ -labelled ranked tree noted $\langle \mathcal{G} \rangle$ and called its *value tree*. This tree is simply obtained by applying an infinite number of times and in a fair way the rewrite rules $\rightarrow_{\mathcal{G}}$ of the HORS \mathcal{G} starting from the start symbol $S \in \mathcal{N}$.

Example 1. Given $\Sigma = \{ \text{if} : 2, \text{data} : 1, \text{Nil} : 0 \}$, consider the HORS \mathcal{G}

$$\begin{cases} \mathbf{S} &= \mathbf{L} \text{ Nil} \\ \mathbf{L} &= \lambda x. \text{if } x \text{ (} \mathbf{L} \text{ (data } x \text{))} \end{cases} \quad (2)$$

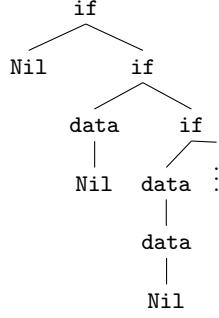
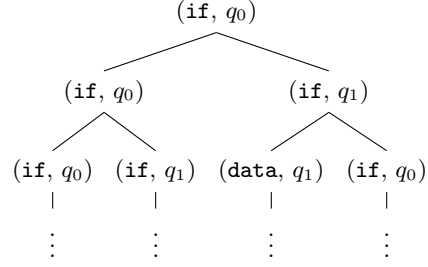
which abstracts a simple program whose function **Main** (abbreviated as **S**) calls a function **Listen** (denoted **L**), starting from an empty list. Depending on a side condition unknown to the user or abstracted by the model-checker, **Listen** either returns a stack of data, or receives a new element and pushes it on the current stack. The value tree $\langle \mathcal{G} \rangle$ of this scheme, depicted in Figure 1, provides an abstraction of the set of potential executions of the program. Note that even though the program **Main** is very simple, its execution tree $\langle \mathcal{G} \rangle$ is not regular, since it admits an infinite number of different subtrees. This justifies from a practical point of view to study how traditional model-checking techniques could be adapted to the HORS \mathcal{G} .

λ -calculus with recursion. It is well-known among the specialists of the λ -calculus that higher-order recursion schemes can be nicely represented as simply-typed λ -terms in a λ -calculus extended with a fixpoint operator Y . The resulting λY -calculus is thus defined by adding to the simply-typed λ -calculus, a fixpoint operator Y_{σ} of type $\sigma \rightarrow \sigma$ together with a rewriting rule

$$Y_{\sigma} M \rightarrow_{\delta} M (Y_{\sigma} M)$$

for every simple type σ .

Proposition 1. *For every HORS \mathcal{G} of ranked alphabet Σ , there exists a closed λY -term $t :: o$ with constants in Σ , such that the λY -term t converges to the*

**Fig. 1:** An order-1 value tree.**Fig. 2:** An APT run-tree.

value-tree $\langle \mathcal{G} \rangle$ in the traditional sense of Böhm trees in the λY -calculus. Conversely, there exists for every closed λY -term $t :: o$ with constants in Σ a HORS \mathcal{G} of same ranked alphabet Σ , such that the λY -term t converges to $\langle \mathcal{G} \rangle$.

An important benefit of this equivalence property is that the λY -calculus is very well understood from the semantic point of view, and thus somewhat simpler to study mathematically speaking than higher-order recursion schemes.

3 MSO and alternating parity tree automata

As explained in the introduction, there is a beautiful correspondence between the formulas of monadic second-order logic (MSO) and alternating parity tree automata, which we briefly recall here for the sake of completeness.

Proposition 2. *For every ranked alphabet Σ , one has the following equivalence:*

- Every MSO formula φ over Σ -labelled trees can be translated to an APT \mathcal{A}_φ of same ranked alphabet Σ , such that φ holds at the root of a Σ -labelled tree T iff \mathcal{A}_φ has an accepting run-tree over T from its initial state q_0 .
- Conversely, every APT \mathcal{A} of ranked alphabet Σ can be translated to a MSO formula $\varphi_{\mathcal{A}}$ of same ranked alphabet, such that for every Σ -labelled tree T , \mathcal{A} has an accepting run-tree over T from its initial state q_0 if and only if the MSO-formula $\varphi_{\mathcal{A}}$ holds at the root of T .

Recall that alternating parity tree automata (APT) are non-deterministic top-down tree automata with the additional ability to *duplicate* or to *erase* subtrees. Typical transitions are thus of the form

$$\delta(q_0, \text{if}) = (2, q_0) \wedge (2, q_1) \quad \delta(q_1, \text{if}) = (1, q_1) \wedge (2, q_0) \quad (3)$$

When a node labelled with **if** is visited in state q_0 , its left subtree is “dropped” or “erased” while the right one is “explored twice” or “duplicated”, with q_0 as initial state in one copy, and q_1 as initial state in the other copy. The second transition

does not use alternation, and would be usually written as $(q_1, \text{if}, q_1, q_0) \in \Delta$ in a nondeterministic tree automaton. Run-trees of an alternating parity tree automaton are unranked, and their shape may differ a lot from the original tree. The effect of the transitions (3) over the tree of Figure 1 is depicted in Figure 2. In general, a transition is of the shape

$$\delta(q, a) = \bigvee_{i \in I} \bigwedge_{j \in J_i} (d_{i,j}, q_{i,j}) = \bigvee_{i \in I} \varphi_i \quad (4)$$

where the union stands for non-determinism, and the conjunction for alternation: after i is chosen, for every $j \in J_i$, the automaton runs with state $q_{i,j}$ over a copy of its subtree in direction $d_{i,j}$. For every i , we say that φ_i is a *conjunctive clause* of the formula $\delta(q, a)$.

Seen from an automata-theoretic point of view, monadic second-order (MSO) logic is equivalent to the modal μ -calculus. As such it enables one to express safety properties (typically, that a given state “error” is never encountered) as well as liveness properties (typically, that a given state “happy” is visited infinitely often). The safety properties are inductive: it is enough to check that no finite approximation of a computation enters an error state, while the liveness properties are coinductive, since they specify infinitary behaviors. Moreover, MSO logic and the modal μ -calculus are sufficiently expressive to alternate these inductive and coinductive specifications. This alternation is handled by extending APT with a *parity condition* over their run-trees. Alternating parity automata are thus equipped with a *coloring function* $\Omega : Q \rightarrow \mathbb{N}$, which associates a color to each state q of the automaton. This coloring of the states $q \in Q$ of the automaton induces a coloring of the nodes of its run-trees, in the expected way. Following the principles of parity games, an infinite branch of such a run-tree is declared winning when the greatest color occurring infinitely often in it is even. A run-tree of the automaton is then accepted precisely when all its infinite branches are winning. In the sequel, we find convenient to consider the set $Col = \Omega(Q) \uplus \{\epsilon\}$ of colors appearing in the alternating parity automaton \mathcal{A} under study. The extra color ϵ is added as a neutral color, in order to reflect the comonadic nature of colors, as we will explain in the later §4. The following definition will also be useful in the sequel, in order to connect the alternating parity automaton \mathcal{A} and the finitary semantics of linear logic:

Definition 1. *Given a state $q \in Q$ and a n -ary constructor $a \in \Sigma$, we say that a n -tuple $\alpha \in (\mathcal{P}_{fin}(Col \times Q))^n$ satisfies the formula $\delta(q, a)$ when α is of the form*

$$\alpha = (\{ (c_{1i_1}, q_{1i_1}) \mid i_1 \in I_1 \} , \dots , \{ (c_{ni_n}, q_{ni_n}) \mid i_n \in I_n \})$$

and there exists a n -tuple of subsets $J_1 \subseteq I_1, \dots, J_n \subseteq I_n$ such that

$$\bigwedge_{k=1}^n \bigwedge_{j_k \in J_k} (k, q_{kj_k}) \quad (5)$$

defines a conjunctive clause of the formula $\delta(q, a)$, and such that moreover

$$\forall k \in \{1, \dots, n\} \quad \forall j \in J_k \quad c_{kj} = \Omega(q_{kj}).$$

In other words, α is a n -tuple of sets $\{(c_{1i_k}, q_{1i_k}) \mid i_k \in I_k\}$ of states annotated with colors, each of them corresponding to one of the n subtrees below the symbol a . Moreover, each such set should contain a subset $\{(\Omega(q_{1i_k}), q_{1i_k}) \mid i_k \in J_k\}$ of appropriately colored states, such that (5) defines a conjunctive clause of the formula $\delta(q, a)$. The general idea is that the n -tuple is allowed to contain more colored states than what is strictly required for the transition $\delta(q, a)$ to be performed by the alternating parity automaton \mathcal{A} . This definition will be crucial in the construction of the finitary semantics which, we will see, is based on downward-closed sets and subtyping.

4 The Scott semantics of linear logic

Here, we adapt the infinitary and colored relational semantics of linear logic formulated in [4,5] to the finitary Scott semantics, where formulas of linear logic are interpreted as partial orders. The semantics of linear logic is *qualitative* in the technical sense that its exponential modality $!$ is interpreted using the finite *powerset* construction, which transports finite sets into finite sets, in contrast to the finite multiset construction used in the traditional and *quantitative* relational semantics. The terminology of Scott semantics comes from the fact that in the derived semantics of the simply-typed λ -calculus, every type is interpreted as a prime algebraic complete lattice, and every simply-typed λ -term as a Scott-continuous function. So, let **ScottL** denote the category with preorders $\mathbf{A} = (A, \leq_A)$ as objects and downward-closed binary relations $R \subseteq A \times B$ as morphisms $(A, \leq_A) \rightarrow (B, \leq_B)$. Here, by a downward-closed relation, we mean a binary relation R such that for all $a, a' \in A$ and $b, b' \in B$, one has :

$$(a, b) \in R \text{ and } a \leq_A a' \text{ and } b' \leq_B b \quad \Rightarrow \quad (a', b') \in R.$$

The binary relation R is thus downward closed in the partial order $(A, \leq_A)^{op} \times (B, \leq_B)$ interpreting the formula $(A, \leq_A) \multimap (B, \leq_B)$ in the Scott semantics. The intuition guiding this property is that if a binary relation R interpreting a proof of linear logic can produce an output b from an input a , then the same binary relation can also produce a less informative output b' from a more informative input a' . It is well-known in the literature on linear logic that this “saturation property” is essential in order to obtain a relational semantics of linear logic with a qualitative (that is, based on finite sets instead of finite multisets) interpretation of the exponential modality. This remark is generally attributed to Ehrhard, see [8] for details. The composition in **ScottL** is relational, since relational composition preserves the property of being downward-closed. The identity morphism over (A, \leq_A) is

$$id_A = \{(a', a) \mid a \leq_A a'\}$$

ScottL is a compact closed category with products, with

$$\begin{aligned} (A, \leq_A) \otimes (B, \leq_B) &= (A \times B, \leq_A \times \leq_B) & 1 &= (\{\star\}, =) \\ (A, \leq_A) \& (B, \leq_B) &= (A \uplus B, \leq_A \uplus \leq_B) & \top &= (\emptyset, \emptyset) \\ (A, \leq_A)^\perp &= (A, \geq_A) \end{aligned}$$

The exponential modality

$$! : A \mapsto !A : \mathbf{ScottL} \longrightarrow \mathbf{ScottL}$$

is then defined by associating to the ordered set (A, \leq_A) the set $\mathcal{P}_{fin}(A)$ of finite subsets of A , where two finite subsets u and v are ordered in the following way:

$$u \leq_{!A} v \iff \forall a \in u, \exists b \in v, u \leq_A v.$$

Recall that the endofunctor $!$ transports every morphism $R : A \rightarrow B$ of the category \mathbf{ScottL} to the following morphism:

$$!R = \{ (u, v) \in !A \times !B \mid \forall b \in v \exists a \in u (a, b) \in R \} : !A \rightarrow !B$$

The endofunctor $!$ is in fact a comonad and defines a Seely category, and thus a model of full propositional linear logic, based on the category \mathbf{ScottL} , see for instance [11].

The coloring comonad. As we have shown in [4,5], the treatment of colors by alternating parity automata follows essentially the same comonadic principles as the treatment of copies in linear logic. This connection between higher-order model checking and linear logic leads to a coloring comonad \square on the relational semantics of linear logic, which we adapt here to the qualitative Scott semantics. To that purpose, we fix a finite set of colors Col containing a neutral element ϵ , and consider the coloring function $Q \rightarrow Col$ which associates a color to every state of a parity tree automaton \mathcal{A} , see the previous discussion in §3. The modality \square is then defined in the following way for an ordered set (A, \leq_A) and a morphism $R : (A, \leq_A) \rightarrow (B, \leq_B)$:

$$\begin{aligned} \square(A, \leq_A) &= (A, \leq_A) \& \cdots \& (A, \leq_A) \\ &\cong (\{(i, a) \mid i \in Col, a \in A\}, \leq_{\square A}) \\ (i, a) \square R (j, b) &\text{ iff } i = j \text{ and } a R b \end{aligned}$$

where $(i, a) \leq_{\square A} (j, a')$ iff $i = j$ and $a \leq_A a'$. The comonadic structure of \square is provided by the following structural morphisms

$$\begin{aligned} \mathbf{dig}_A &= \{((\max(c_1, c_2), a), (c_1, (c_2, a')))) \mid a' \leq_A a\} & : \square A \rightarrow \square \square A \\ \mathbf{der}_A &= \{((\epsilon, a), a') \mid a' \leq_A a\} & : \square A \rightarrow A \\ m_{A,B} &= \{(((i, a), (i, b)), ((i, (a', b')))) \mid a' \leq_A a, b' \leq_B b\} & : \square A \otimes \square B \rightarrow \square(A \otimes B) \\ m_1 &= \{(\star, (c, \star)) \mid c \in Col\} & : 1 \rightarrow \square 1 \end{aligned}$$

As we did in the case of the relational semantics [4,5], we define a distributive law $\lambda : ! \circ \square \Rightarrow \square \circ !$ between the comonads $!$ and \square defined as the natural transformation:

$$\lambda_A = \{(\{(c_j, a'_j)\}, (c, \{a_i\})) \mid \forall i \exists j \ c = c_j \text{ and } a_i \leq_A a'_j\} : !\square A \rightarrow \square !A$$

The existence of such a distributive law λ enables us to equip the composite functor $\mathbf{!} = ! \circ \square$ with a comonadic structure. It appears moreover that this colored exponential functor $\mathbf{!}$ satisfies the axioms of a Seely category, and thus defines a model of full propositional linear logic. We denote by $\mathbf{ScottL}_{\mathbf{!}}$ its Kleisli category.

5 A finitary interpretation of the simply-typed λ -calculus

In order to simplify the discussion, we suppose given an alternating parity tree \mathcal{A} over a signature Σ , with set of states Q and with transition function δ . As a Kleisli category associated to a model of linear logic, the category \mathbf{ScottL}_\sharp is cartesian closed and thus a model of the simply-typed λ -calculus. The simple types are interpreted inductively as

$$\llbracket \sigma \rightarrow \tau \rrbracket = \sharp \llbracket \sigma \rrbracket \multimap \llbracket \tau \rrbracket \quad \text{and} \quad \llbracket o \rrbracket = \perp\!\!\!\perp = (Q, =)$$

The interpretation of the simply-typed λ -terms is standard, except for the interpretation of the elements of the ranked alphabet Σ , seen as here constants of the simply-typed λ -calculus, which are interpreted as follows:

$$\llbracket a \rrbracket_{\mathcal{A}} = \{ (\alpha, q) \mid q \in Q \text{ and } \alpha \text{ satisfies the formula } \delta(q, a) \}$$

As explained in [5] in the case of the quantitative relational semantics of linear logic, this interpretation of the elements of Σ corresponds to a Church encoding of the alternating parity automaton \mathcal{A} , encoded in the present case in the qualitative Scott semantics of linear logic.

Example 2. Recall the two transitions (3) introduced as running example in §3:

$$\delta(q_0, \text{if}) = (2, q_0) \wedge (2, q_1) \quad \delta(q_1, \text{if}) = (1, q_1) \wedge (2, q_0)$$

Setting $c_i = \Omega(q_i)$, these transitions imply that

$$(u_1, u_2, q_0) \in \llbracket \text{if} \rrbracket_{\mathcal{A}} \quad \text{and} \quad (v_1, v_2, q_1) \in \llbracket \text{if} \rrbracket_{\mathcal{A}}$$

for every finite sets $u_1, u_2, v_1, v_2 \in \sharp \perp\!\!\!\perp = \mathcal{P}_{fin}(Col \times Q)$ satisfying moreover that $\{(c_0, q_0), (c_1, q_1)\} \subseteq u_2$, that $(c_1, q_1) \in v_1$ and that $(c_0, q_0) \in v_2$.

Using these interpretations in \mathbf{ScottL} of the elements of the ranked alphabet Σ , we construct the interpretation

$$\llbracket \Gamma \vdash t :: \tau \rrbracket_{\mathcal{A}} \subseteq (\sharp \llbracket \sigma_1 \rrbracket \otimes \cdots \otimes \sharp \llbracket \sigma_n \rrbracket) \multimap \llbracket \tau \rrbracket$$

of any λ -term t of type τ in a context of typed variables Γ , with constants in the ranked alphabet Σ . An alternative way to describe this interpretation is to express it as an intersection type system with subtyping, in the style of Coppo, Dezani, Honsell and Longo [3] and more recently Terui [11] in the framework of linear logic. In this formulation, sequents are of the following form

$$\Gamma = x_1 : u_1 :: \sigma_1, \dots, x_n : u_n :: \sigma_n \vdash t : \alpha :: \tau$$

where $u_i \in \sharp \llbracket \sigma_i \rrbracket$ and $\alpha \in \llbracket \tau \rrbracket$. The typing rules are presented in Figure 4, with the subtyping relation \leq_A defined inductively in Figure 3. Note that the coloring $\square_c \Gamma$ of a context is defined inductively as

$$\begin{aligned} \square_c (x : u :: \sigma, \Gamma) &= x : \square_c u :: \sigma, \square_c \Gamma \\ \square_c \{ (c_i, \alpha_i) \} &= \{ (\max(c, c_i), \alpha_i) \} \end{aligned}$$

Proposition 3. *The sequent*

$$\Gamma = x_1 : u_1 :: \sigma_1, \dots, x_n : u_n :: \sigma_n \vdash t : \alpha :: \tau$$

$$\frac{}{q \leq_{\perp} q} \quad \frac{\forall (c, \alpha) \in u \quad \exists (c, \beta) \in v \quad \alpha \leq_A \beta}{u \leq_{\sharp A} v} \quad \frac{v \leq_{\sharp A} u \quad \alpha \leq_B \beta}{u \rightarrow \alpha \leq_{\sharp A \multimap B} v \rightarrow \beta}$$

Fig. 3: Inference rules for the preorders associated with simple types.

$$\begin{array}{c} \text{Ax} \quad \frac{\exists (\epsilon, \alpha') \in u \quad \alpha \leq_{\llbracket \sigma \rrbracket} \alpha'}{x : u :: \sigma \vdash x : \alpha :: \sigma} \quad \delta \quad \frac{q \in Q \text{ and } \alpha \text{ satisfies } \delta(q, a)}{\emptyset \vdash a : \alpha \rightarrow q :: \sigma} \quad \lambda \quad \frac{\Gamma, x : u :: \sigma \vdash M : \alpha :: \tau}{\Gamma \vdash \lambda x. M : u \rightarrow \alpha :: \sigma \rightarrow \tau} \\ \\ \text{App} \quad \frac{\Gamma_0 \vdash M : \{(c_1, \beta_1), \dots, (c_n, \beta_n)\} \rightarrow \alpha :: \sigma \rightarrow \tau \quad \Gamma_i \vdash N : \beta_i :: \sigma \quad (\text{for all } i)}{\Gamma_0 \cup \square_{c_1} \Gamma_1 \cup \dots \cup \square_{c_n} \Gamma_n \vdash M N : \alpha :: \tau} \end{array}$$

Fig. 4: Type-theoretic computation of denotations in **ScottL_‡**

is provable in this intersection type system if and only if

$$(u_1, \dots, u_n, \alpha) \in \llbracket \Gamma \vdash t :: \tau \rrbracket_A \subseteq (\sharp \llbracket \sigma_1 \rrbracket \otimes \dots \otimes \sharp \llbracket \sigma_n \rrbracket) \multimap \llbracket \tau \rrbracket$$

6 The recursion operator **Y**

At this stage, we are ready to shift from the colored semantics of the simply-typed λ -calculus formulated in §5 to a colored semantics of the simply-typed λY -calculus. To that purpose, we construct a Conway operator **Y** in the category **FinScottL** defined as the full subcategory of **ScottL** consisting of the *finite* ordered sets. Note that **FinScottL** defines a Seely category, and thus a model of full propositional linear logic. The Conway operator **Y** is defined a family of operations $\mathbf{Y}_{X,A}$ transporting a binary downward-closed relation

$$R : \sharp X \otimes \sharp A \multimap A$$

into a binary downward-closed relation

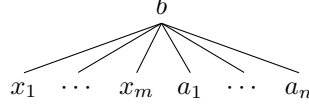
$$\mathbf{Y}_{X,A}(R) : \sharp X \multimap A$$

and satisfying a series of conditions originally stated by Bloom and Esik [1] in cartesian closed categories, and adapted in [4] to the particular framework of Seely categories. Note that, such a Conway operator on **FinScottL** defines a Conway operator in the sense of [1] in the cartesian-closed category **FinScottL_‡**. Just as in the case of the relational semantics, see [4] for details, the important point here is that the colors added to the original Scott semantics will enable us to alternate least and greatest fixpoints (and thus inductive and coinductive reasoning) in the definition of the fixpoint operator **Y**, using the appropriate parity condition.

Semantic run-trees. Given a relation $R : \sharp X \otimes \sharp A \multimap A$ and $a \in A$, we define the set $\mathbf{comp}(R, a)$ of *semantic run-trees* of R producing $a \in A$ as the set of possibly infinite $(X \uplus A)$ -labelled trees, with nodes colored by elements of Col , and such that the four conditions below are satisfied:

1. the root of the tree is labelled by a , and has neutral color ϵ ,

2. the inner nodes of the tree are labelled by elements of the set A ,
3. the leaves are labelled by elements of the set $X \uplus A$,
4. for every node labelled by an element $b \in A$:
 - if b is an inner node, letting a_1, \dots, a_n denote the labels of its children belonging to A and x_1, \dots, x_m the labels belonging to X :



and letting c_i (resp. d_j) be the color of the node labelled x_i (resp. a_j),

$$(\{(c_1, x_1), \dots, (c_m, x_m)\}, \{(d_1, a_1), \dots, (d_n, a_n)\}, b) \in R$$

- if b is a leaf, then $(\emptyset, \emptyset, b) \in R$.

At this point, we adapt to semantic run-trees the usual acceptance condition on the run-trees of an alternating parity automata: an infinite branch of the semantic run-tree is winning if and only if an element of $Col \setminus \{\epsilon\}$ occurs infinitely often along it, and if the maximal such element is even. A semantic run-tree is declared winning if and only if all its infinite branches are.

Given a semantic run-tree *witness*, we define the set $\mathbf{leaves}(witness) \subseteq \downarrow X$ as the set of elements (c, x) where (c', x) is a leaf of *witness* labelled with $x \in X$, and c is the maximal color encountered on the path from the leaf to the root or *witness*.

Fixpoint operator. We now define the fixpoint of a binary relation

$$R : \downarrow X \otimes \downarrow A \multimap A$$

as the downward-closed binary relation

$$\mathbf{Y}_{X,A}(R) = \{ (u, a) \mid \exists witness \in \mathbf{comp}(R, a) \text{ with } u = \mathbf{leaves}(witness) \text{ and } witness \text{ is a winning semantic run-tree.} \} \quad (6)$$

Proposition 4. *The fixpoint operator \mathbf{Y} is a Conway operator over $\mathbf{FinScottL}$. Consequently, its Kleisli category $\mathbf{FinScottL}_\downarrow$ is a model of the λY -calculus.*

As in §5, we find useful and even illuminating to formulate a type-theoretic counterpart to our definition of the Conway operator $\mathbf{Y}_{X,A}$ provided by the following typing rule Y_σ which should be added to the original type system of Figure 4 :

$$Y_\sigma \frac{\Gamma_0 \vdash M : \{(c_1, \beta_1), \dots, (c_n, \beta_n)\} \rightarrow \alpha :: \sigma \rightarrow \sigma \quad \Gamma_i \vdash Y_\sigma M : \beta_i :: \sigma}{\Gamma_0 \cup \square_{c_1} \Gamma_1 \cup \dots \cup \square_{c_n} \Gamma_n \vdash Y_\sigma M : \alpha :: \sigma}$$

In the resulting intersection type system, derivations of infinite depth are allowed, and have colored nodes, defined as follows:

- for every occurrence of the rule Y_σ , we assign color c_i to the node $\Gamma_i \vdash Y_\sigma M : \beta_i :: \sigma$.
- all the other nodes are assigned the neutral color ϵ .

An infinite derivation tree is then accepted as a proof of the system when all its branches are winning, in the same sense as for the branches of a semantic run-tree.

Theorem 1. *Given a λY -term t , the sequent*

$$\Gamma = x_1 : u_1 :: \sigma_1, \dots, x_n : u_n :: \sigma_n \vdash t : \alpha :: \tau$$

has a winning derivation tree in the type system with fixpoints iff

$$(u_1, \dots, u_n, \alpha) \in \llbracket \Gamma \vdash t :: \tau \rrbracket_{\mathcal{A}} \subseteq (\downarrow \llbracket \sigma_1 \rrbracket \otimes \dots \otimes \downarrow \llbracket \sigma_n \rrbracket) \multimap \llbracket \tau \rrbracket$$

At this point, we take advantage of the correspondence recalled in Proposition 1 between higher-order recursion schemes (HORS) on the ranked alphabet Σ , and closed λY -terms with constants in the same alphabet Σ . Indeed, the correspondence enables us to justify the following typing rule for HORS :

$$\frac{\Gamma_0, F : \{(c_1, \beta_1), \dots, (c_n, \beta_n)\} :: \sigma \vdash \mathcal{R}(F) : \alpha :: \sigma \quad \Gamma_i \vdash F : \beta_i :: \sigma \quad (\forall i)}{\Gamma_0 \cup \square_{c_1} \Gamma_1 \cup \dots \cup \square_{c_n} \Gamma_n \vdash F : \alpha :: \sigma}$$

which provides a direct mean to type the HORS \mathcal{G} in the intersection type system, in such a way as to reflect its interpretation $\llbracket \mathcal{G} \rrbracket_{\mathcal{A}} \subseteq Q$ in the Scott semantics.

7 Finitary semantics solve the selection problem

The first theorem of the section establishes a perfect correspondence between our finitary interpretation $\llbracket \mathcal{G} \rrbracket_{\mathcal{A}}$ of the higher-order recursion scheme \mathcal{G} in the Scott semantics, and the set of accepting states of the automaton \mathcal{A} :

Theorem 2. *An alternating parity tree automaton \mathcal{A} has an accepting run-tree with initial state q_0 over the value tree $\langle \mathcal{G} \rangle$ of a higher-order recursion scheme \mathcal{G} if and only if $q_0 \in \llbracket \mathcal{G} \rrbracket_{\mathcal{A}}$.*

By Theorem 1, checking whether $q_0 \in \llbracket \mathcal{G} \rrbracket_{\mathcal{A}}$ is equivalent to checking whether there exists a derivation of the sequent $\emptyset \vdash S : q_0 :: o$ in the colored intersection type system defined in §6. Since the interpretation of simple types in **FinScottL** is finite, only finitely many intersection types and contexts may occur in such a derivation. Hence, searching for a derivation of the sequent $\emptyset \vdash S : q_0 :: o$ reduces in this case to solving a finite parity game whose nodes are precisely the sequents of the derivation tree. This has the following immediate consequence:

Corollary 1. *The local model-checking problem is decidable.*

Recall moreover that the existence of a winning strategy in a finite parity game implies that there exists a *memoryless* winning strategy. In this setting, winning strategies correspond to winning derivation trees of the intersection type system, and memoryless strategies correspond to derivation trees admitting a finite representation using backtracking pointers. From such a finite representation π , one can define a higher-order recursion scheme \mathcal{G}_π on a ranked

alphabet $\Sigma_{\mathcal{A}}$ obtained from Σ by annotating every terminal a with elements of its interpretation $\llbracket a \rrbracket_{\mathcal{A}}$. The HORS \mathcal{G}_q has a non-terminal $F_{\alpha}(o)$ for every occurrence o of the non-terminal F in the finite representation π of the derivation tree, where α is the intersection type of the occurrence o of F in π . Each occurrence o of a non-terminal F then induces a rewrite rule $F_{\alpha}(o) \rightarrow_{\mathcal{G}_q} \text{term}(o)$ where $\text{term}(o)$ is an annotated version of the λ -term $\mathcal{R}(F)$ coming from the original scheme \mathcal{G} . The annotation of $\text{term}(o)$ is obtained by annotating the non-terminals and the terminals of $\mathcal{R}(F)$ with the intersection types occurring in the finite representation π of the derivation tree. This defines a higher-order recursion scheme \mathcal{G}_q , which generates a run-tree $\langle \mathcal{G}_q \rangle$ of the alternating parity tree automaton \mathcal{A} over $\langle \mathcal{G} \rangle$. As a consequence:

Theorem 3. *The selection problem is decidable.*

8 Conclusions and perspectives

In this paper, we explain how to apply our semantic approach to higher-order model-checking based on linear logic, in order to establish the decidability of local model-checking and of the selection problem. Our approach provides a rigorous and compositional approach to higher-order model-checking, and adapts to the inductive-coinductive framework of MSO logic a nice and well-established connection between linear logic, Scott domains, and intersection types. Future work includes a detailed comparison with a similar line of work on finite models of the λY -calculus currently developed by Salvati and Walukiewicz [10].

References

1. Bloom, S.L., Ésik, Z.: Fixed-point operations on ccc's. part i. TCS 155 (1996)
2. Carayol, A., Serre, O.: Collapsible pushdown automata and labeled recursion schemes: Equivalence, safety and effective selection. In: LICS (2012)
3. Coppo, M., Dezani-Ciancaglini, M., Honsell, F., Longo, G.: Extended Type Structures and Filter Lambda Models. In: Logic Colloquium 82 (1984)
4. Grellois, C., Melliès, P.: An infinitary model of linear logic. In: Pitts, A.M. (ed.) FoSSaCS. LNCS, vol. 9034 (2015)
5. Grellois, C., Melliès, P.: Relational semantics of linear logic and higher-order model-checking. submitted, <http://arxiv.org/abs/1501.04789> (2015)
6. Haddad, A.: Shape-preserving transformations of higher-order recursion schemes. Ph.D. thesis, Université Paris Diderot (2013)
7. Knapik, T., Niwinski, D., Urzyczyn, P.: Higher-order pushdown trees are easy. In: Nielsen, M., Engberg, U. (eds.) FoSSaCS. LNCS, vol. 2303 (2002)
8. Melliès, P.A.: Categorical semantics of linear logic. In: Interactive models of computation and program behaviour, pp. 1–196 (2009)
9. Ong, C.H.L.: On model-checking trees generated by higher-order recursion schemes. In: LICS, pp. 81–90. IEEE Computer Society (2006)
10. Salvati, S., Walukiewicz, I.: A model for behavioural properties of higher-order programs. Personal communication
11. Terui, K.: Semantic evaluation, intersection types and complexity of simply typed lambda calculus. In: RTA (2012)